EXOMARS 2020 SVF: Re-use of SIMULUS Models on TSIM-based Simulator

Juan-Rafael García-Blanco[(1)], David González-Arjona[(1)], Carmelo Carrascosa-Sanz[(1)], Mónica Rollán-Galindo[(1)]

GMV[(1)]
Calle Isaac Newton 11,Tres Cantos (Madrid), 28760 Spain
E-mail: jrblanco@gmv.com, dgarjona@gmv.com, ccarrascosa@gmv.com, mrollan@gmv.com

ABSTRACT

The EXOMARS 2020 SVF contains software models of all On-Board Computer units of the spacecraft. The development of this kind of models is a difficult task because of the great level of representativeness they need to achieve. These software models have to support the execution of the actual On-Board Software of the mission, which is also validated in a real hardware environment.

The On-Board Computer of EXOMARS 2020 is similar to the one already mounted in Spanish spacecraft SEOSAT, for which GMV has developed models of its On-Board Computer units. These models have been created using ESA's SIMULUS suite of tools and frameworks, which are compliant with the SMP2 standard. This standard proposes mechanisms and guidelines that enable portability of models between simulators.

The models utilised in the SEOSAT Operational Simulator have been ported to the EXOMARS 2020 SVF to reduce development effort and mitigate risks. This activity has allowed GMV to assess the convenience of following the SMP2 standard for the development of reusable components.

The EXOMARS 2020 SVF environment greatly differs from the one provided by ESA's SIMULUS:

- It is built with Visual Studio C++ and runs on a Windows system.
- It uses TSIM to provide basic simulation services and LEON3 processor emulation.
- It does not comply with SMP2. Communication with other SVF components is performed through pipes.

Porting of existing SMP2 models to this new environment required the development of different components. These components adapt the TSIM-based environment to expose the standard SMP2 services used by the models.

The resulting On-Board Computer simulator has been validated using EXOMARS 2020 specific drivers. This validation activity has also served to discover small discrepancies in the behaviour of On-Board Computer models, whose implementation is driven by the need of supporting a specific On-Board Software.

This paper presents the successful process of porting existing SMP2 models to the EXOMARS 2020 SVF environment. It describes the newly developed components, the required modifications to existing models, and the limitations faced in a TSIM-based platform. Enhancement options for the SVF environment are also explored.

INTRODUCTION

Software Verification Facilities (SVF) integrate software models of the different units in the On-Board Computer (OBC), including a processor emulator that executes the On-Board Software (OBSW) under verification. These models need to resemble closely the real hardware in terms of both functionality and timing. The development of this kind of software

requires of considerable effort, mainly because the hardware units are complex by themselves, and also because the manufacturer's documentation does not contain detailed information on how units work internally. Therefore, model re-use is an advantage when creating a SVF for a new mission.

The EXOMARS 2020 OBC integrates Airbus' SCOC3 System-On-Chip (SoC), which contains a single-core LEON3-FT processor and implements usual functions, e.g., TM/TC units, various I/O interfaces, etc. This SoC is also utilised in the Spanish mission SEOSAT, for which GMV has developed the operational simulator for training operators. This simulator includes models of all spacecraft units, including the OBC.

GMV has developed the SEOSAT operational simulator following the SMP2 (Simulation Model Portability) standard [1], and ESOC's SIMULUS infrastructure [2]. The SMP2 standard promotes model portability between simulators and platforms. It achieves portability by defining various aspects of simulators, the most important being the following:

- Clean design architecture. Interaction between models is based on well-defined interfaces, which minimise coupling between simulator components. SMP2 defines various types of model interaction: aggregation, composition, events, data flow. Model communication occurs, therefore, through standard interfaces.
- Mandatory services. Models can access common services, e.g., logging, scheduler, etc. through their corresponding standard interfaces. This minimises coupling with the host platform.

The SIMULUS infrastructure is currently based on SMP2 and provides the following main components:

- SIMSAT soft-real-time simulation kernel. This element acts as the execution platform for simulation models. It implements the SMP2 mandatory and optional services.
- Reference Architecture (REFA). This element defines the software architecture for models of different common spacecraft units and services. Its purpose is to facilitate model portability by ensuring all realisations adhere to defined interfaces.
- Generic Models (GENM). This element comprises SMP2 models of common spacecraft units that are ready to be integrated in simulators. They are used either directly or by extending them to fit the needs of a particular mission.

GMV has ported the same OBC models utilised in the SEOSAT operational simulator to the EXOMARS 2020 SVF, thus saving development and validation effort. These models have not been modified in the porting activity because GMV adapted the EXOMARS 2020 SVF environment to allow execution of SMP2 models. Additional software models have also been developed for the units that are found only in the EXOMARS 2020 OBC.

ENVIRONMENT DESCRIPTION

The development and operating environments of the SEOSAT operational simulator and the EXOMARS 2020 SVF differ considerably. The main difference lies in the support for the SMP2 standard: the EXOMARS 2020 SVF environment has been designed with no standardisation as goal.

Original Environment

The SIMULUS infrastructure officially supports SUSE Linux Enterprise Server (SLES) version 11 x86_64, although it might execute without problems on other operating systems and architectures. The SEOSAT simulator has been built for SLES11 x86_64, utilising all frameworks and tools provided by SIMULUS, and in accordance with the SMP2 standard.

Fig. 1 shows the different technologies utilised in this simulator. Simulator models access only models from the GENM framework, and SMP2 services, which provide isolation from the underlying simulation platform and operating system.

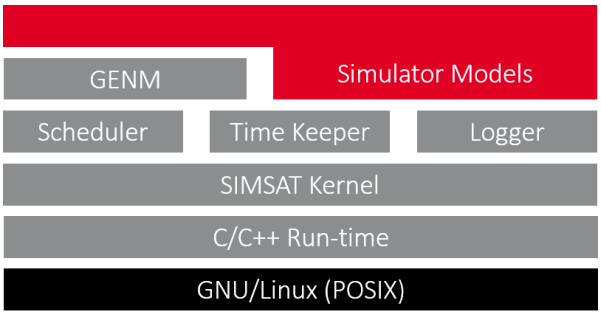

Fig. 1. Technology stack of SEOSAT simulator

The SIMSAT Kernel is in charge of advancing the simulation state by executing the simulator models at their configured frequencies. It implements an adaptation layer between its internal services and the SMP2 standard ones. In this manner, it supports execution of pure SMP2 simulators. The SIMSAT Kernel contains a concurrent scheduler that allows executing models in parallel, but its API is not exposed through SMP2 and, therefore, it is not used in the SEOSAT simulator.

Simulator models, in this case, represent any component in a spacecraft, from a processor emulator to a sensor unit. The simulation platform does not make any distinction between them. They are implemented in the C++ language, as it is officially supported by SMP2. Communication between models is based on direct method calls through custom interfaces.

Target Environment

The EXOMARS 2020 SVF relies on Gaisler's TSIM emulator to provide LEON3-FT emulation capabilities. In addition, it utilises the Windows API (win32), which restricts it to run under the Windows operating system.

The TSIM emulator implements a model of the LEON3 processor, including the Floating-Point Unit (FPU) and Memory Management Unit (MMU), and various peripherals that sit close to the processor, e.g., interrupt controller, memory controller, etc. Since the SPARC architecture has I/O devices mapped into the processor's address space, TSIM allows for custom I/O modules to handle accesses to empty areas of the memory map. It contains, as well, an internal scheduler that advances time according to instruction execution and I/O accesses.

Fig. 2 shows the different technologies and interfaces involved in a TSIM-based simulator. TSIM can execute as a standalone process or linked as a library. It provides a C interface to I/O devices connected to the Amba bus (AHB). This interface allows access to simulation state as well as to emulated devices. In particular, it provides access to TSIM's internal scheduler where user events can be posted.

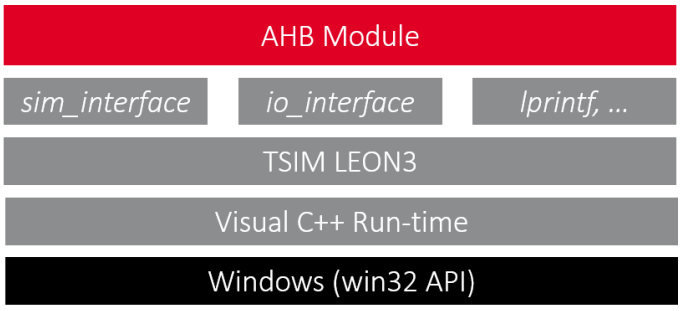

Fig. 2. Technology stack of TSIM simulators

Communication between OBC models and the SVF infrastructure is performed through native pipes. The SVF infrastructure comprises different functions, e.g., TM archiving, TC generation, provision of TM from units connected to system buses, a command prompt for users, etc. This means that control and TM/TC interfaces exists between OBC models and the rest of the SVF.

Comparison

The functionality provided by both environments is very similar, but they greatly differ in three aspects:

- Simulation run-time environment. The SIMSAT Kernel is a pure simulation run-time that provides only services related to simulation; it is independent from the simulated models. On the contrary, TSIM is both a simulation environment and a processor emulator; this means that the processor model cannot be managed like any other model.
- Scheduler. The SIMSAT Kernel implements a soft-real-time scheduler. The TSIM internal scheduler cannot maintain real-time, and it executes events as fast as possible.
- Programming language. The TSIM emulator has a version for VC++ (Visual C++) platforms, which allows compiling and linking C and C++ code together. However, TSIM does not implement neither SMP2 nor any other standard.

ADAPTATIONS TO SUPPORT RE-USED OBC MODELS

The TSIM environment has been improved in various manners to support execution of SMP2 models. Re-used models are written in the C++ language, using only standard libraries. These models have different relations with other models:

- Inheritance. Custom models usually inherit from GENM models that provide basic functionality, e.g., *PoweredUnit* provides on/off status.
- Aggregation. Models access services provided by external models by means of aggregation. This relationship is supported by custom C++ classes in the SMP2 Model Development Kit (MDK).
- Composition. Models can own other models by means of composition. This relationship is also supported by the SMP2 MDK.

The relations with other models translate into dependencies with other software packages that need also be compiled with VC++. All GENM models and the SMP2 MDK are written in the C++ language. The SMP2 base data types are defined in a separate file *Platform.h* that, as distributed with SIMULUS, already contains provisions for Windows platforms. Therefore, the effort dedicated to compile existing models with VC++ is negligible.

Fig. 3 shows the typical structure of SMP2 models. They are isolated from the simulation environment by SMP2 services, which do not exist in the TSIM environment. They communicate with other models through custom or standard (e.g., REFA) interfaces. These are the two major issues addressed during the migration:

- Replication of SMP2 services on top of TSIM, which becomes the simulation kernel.
- Separation of OBC models from the rest of the simulator.
- Replacement of the processor emulator with TSIM.

The effort devoted to adapting the TSIM simulation environment is limited because SIMULUS provides a clean architecture that allows to re-use a notable amount of infrastructure classes, i.e., it has already isolated the native platform aspects.

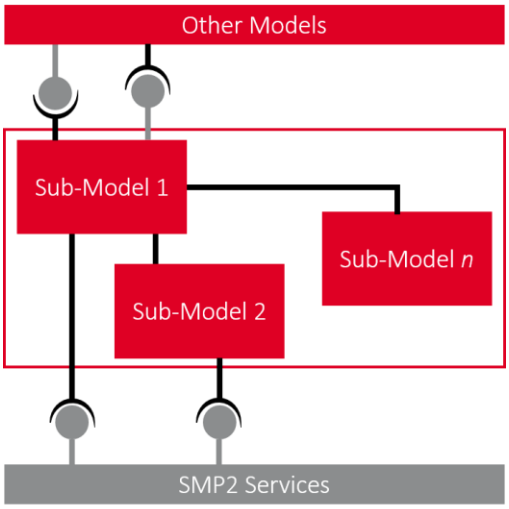

Fig. 3. Structure of SMP2 model

SMP2 Adaptation Layer

The SIMULUS infrastructure contains a Simulator Development Kit (SDK) that supports the development of SMP2 compliant simulators. It has been utilised to provide two basic simulation features:

- Implementation of the SMP2 *ISimulator* interface. This is the single entry point that SMP2 models use to access standard services. The *Sdk::DynamicSimulator* class has been instantiated for this purpose. Its constructor requires passing instances of classes implementing standard service interfaces, i.e., *ILogger*, *ITimeKeeper*, *IScheduler*. Internally, this class accesses host services through the *Sdk::IEnvironment* interface.
- Type registry. Following SMP2 conventions, all data types are registered and described in a global registry; this includes model types. The *Sdk::Publication::TypeRegistry* class has been instantiated for this purpose.
  A type registry is mandatory for porting the existing SEOSAT OBC models to the EXOMARS 2020 SVF because they are configured through standard SMP2 interfaces that can access internal fields of models. This can only be done if models can be introspected, i.e., if all models have registered their fields, along with the associated data types, to the global registry.

Although the SIMULUS SDK includes simple implementations of SMP2 mandatory services, they have not been re-used; custom implementations have been created instead to address the specific aspects of the EXOMARS 2020 SVF. Fig. 4 shows the architecture of the re-implemented SMP2 mandatory services.

The *Logger* service is similar to the one implemented in the SIMULUS SDK but, in this case, messages are transmitted through a native pipe to the SVF infrastructure. For maximum decoupling, message formatting is performed in the *Logger* class, while transmission through native pipes is done in the *LoggerPipeManager* class.

The *Scheduler* service manages an internal queue in the *EmulatorEventManager* class. Each event is assigned a unique identifier and added to TSIM's internal event queue. The implementation of this service adapts the rich SMP2 interface for events to the limited TSIM interface, e.g., TSIM does not support cyclic events.

The *TimeKeeper* service implementation simply accesses the current simulation time provided by TSIM, and adapts it to the units used in SMP2.

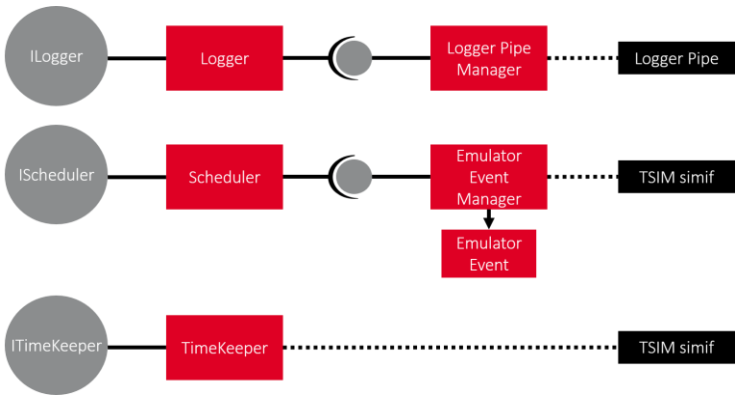

Fig. 4. Implementation of SMP2 services

The mandatory SMP2 services have not been implemented in their entirety; only the parts that are actually needed for OBC models execution have been included.

External Interfaces

While the SEOSAT simulator contains models of the complete spacecraft, only the OBC models are re-used in the EXOMARS 2020 SVF. Partitioning the simulator has been an easy task because it complied with the SMP2 recommendations for the development of re-usable models. Communication between models occurs through well-defined interfaces, making it possible to switch implementations easily. The models outside the OBC boundaries have been replaced with new classes that communicate with the SVF infrastructure using native pipes.

User Interface

The SIMULUS infrastructure includes a dedicated Man-Machine Interface (MMI) program that is able to access methods and the internal state of simulator models. This program is not used in the EXOMARS 2020 SVF. Instead, the TSIM command interface has been used to define a set of user commands that act on simulator models. Only selected functionality is exposed through this interface:

- Set trace level. The GENM's *Entity* model implements a tracing mechanism that assigns an importance value to log messages; at run-time, the model forwards to the *Logger* service only the messages with higher importance than a configured value. This is used mainly for debugging purposes assuming models are verbose enough. Providing access to this feature has been considered fundamental and has been the first command implemented.
- Force Single and Double Event Upset (SEU/DEU). Users also interact with the simulator to inject errors. In particular, these commands simulate the occurrence of a SEU or DEU in the CPU and IO memory controllers.

User interaction is considered one of the weakest points of the EXOMARS 2020 SVF because the SVF infrastructure is not prepared to integrate the powerful simulation platform provided by SMP2.

Integration of Different On-Board Software

The final SEOSAT OBSW image has been utilised during the development of the corresponding operational simulator. Although the OBC models have been designed and implemented based on manufacturer's documentation, executing the real OBSW in the simulator has proved very convenient for validation purposes: it is more complex and thorough than custom unit and integration tests. However, there exists the risk of introducing errors in the modelling if the OBSW does not utilise the hardware to its full extent.

Different tests have been carried out using the official EXOMARS 2020 OBC drivers. These tests have uncovered small errors in the SPW-RMAP unit model inside the SCOC3 SoC. Refinement of this model has been performed during the porting process. This experience has confirmed the need for executing different OBSW images on specific OBC models to assure they are fully validated.

DEVELOPMENT OF NEW MODELS

The tools included in SIMULUS for generation of SMP2 boilerplate code have not been used because of project requirements. Still, new models not present in the SEOSAT OBC have been developed following the SMP2 standard. This approach has been selected because of the following reasons:

- Consistency with the rest of the simulator. In this manner, the complete EXOMARS 2020 OBC conforms to the SMP2 standard and can be later ported to other simulators, e.g., operational simulator.
- It has been found that SMP2 is a good foundation when multiple people work together to develop software. It clearly defines how information is transferred between different components, e.g., aggregation, composition, etc. Thorough use of the SMP2 results in coherent software that can be easily explored and maintained.

CONCLUSIONS

The EXOMARS 2020 SVF activity has proven that SMP2 effectively allows models to be re-used between simulators, even running on different platforms. In addition, it has also made apparent that the modular design of the SIMULUS infrastructure facilitates porting. Re-use has decreased considerably the effort devoted to creating and validating the EXOMARS 2020 OBC simulator.

The SEOSAT OBC models have been ported to a new infrastructure without considerable effort because they are well isolated from the host system and surrounding models through interfaces. The implementation of mandatory SMP2 services that enable existing models to execute has been eased by the existence of the SIMULUS SDK. Validation using specific EXOMARS 2020 OBC drivers has been straightforward, except for small discrepancies found in one single model.

The EXOMARS 2020 OBC simulator is considered a powerful tool for testing and validating software because models can be fully introspected. However, access to the simulation state is mostly hidden because the SVF architecture is not aware of the simulation platform, e.g., it cannot traverse the simulator tree extracting the simulator state, etc. Only the essential functionality has been exposed through TSIM command interface.

Development of new models have been performed following the SMP2 standard. This has resulted in a simulator where all interactions are based on the same mechanisms, which eases maintainability and reusability.

REFERENCES

[1] SMP 2.0 Handbook, EGOS-SIM-GEN-TN-0099 issue 1.2 28-Oct-2005.

[2] User Guide for developing applications depending on SIMULUS, ESA-EGOS-SIM-MAN-0001 issue 1.1 31-Oct-2016.